

Maze (Shortest Path)

Generated by Doxygen 1.12.0

1 README	1
1.1 cosc2804-assignment3-template	1
1.1.1 Team Allocation	1
1.1.2 Team Video Link	1
1.1.3 Team Progress	1
1.1.3.1 Toms's Commits	1
1.1.3.2 Liam's Commits	2
1.1.3.3 Erfan's Commits	2
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Class Documentation	7
4.1 Agent Struct Reference	7
4.2 Block Struct Reference	7
4.2.1 Detailed Description	7
4.3 Maze Class Reference	8
4.4 Place_Maze Class Reference	8
4.4.1 Detailed Description	8
4.4.2 Member Function Documentation	9
4.4.2.1 Build_Maze()	9
4.4.2.2 Clear_Maze()	9
4.4.2.3 Flatten_Terrain()	9
4.4.2.4 Get_Player_Build_Pos()	9
4.4.2.5 Has_Terrain()	9
4.4.2.6 Load_Maze()	9
4.4.2.7 Place_Carpet()	9
4.4.2.8 Print_Maze_Size()	10
4.4.2.9 Restore_Terrain()	10
4.4.2.10 Save_Terrain()	10
4.4.2.11 Set_Player_Build_Pos()	10
4.5 Solve_Maze Class Reference	10
4.6 Terrain_Array Class Reference	11
4.6.1 Detailed Description	11
4.6.2 Member Function Documentation	11
4.6.2.1 add_block()	11
4.6.2.2 clear()	11
4.6.2.3 get_block()	11
4.6.2.4 print()	12
4.6.2.5 size()	12

5 File Documentation	13
5.1 Agent.h	13
5.2 Block.h	13
5.3 Maze.h	13
5.4 menuUtils.h	14
5.5 Place_Maze.h	16
5.6 Solve_Maze.h	17
5.7 Terrain_Array.h	18
Index	19

Chapter 1

README

1.1 cosc2804-assignment3-template

1.1.1 Team Allocation

	Member 1 (Tom Castanelli) [s4072172]	Member 2 (Liam Moore) [s4095280]	Member 3 (Erfan Samandarian) [s4089117]
Testing	Test to cover Member 1's and Member 2's work	Test to cover Member 2's and Member 3's work	Test to cover Member 3's and Member 1's work
Base Program	Maze Generation	Maze Solving	Build maze & Cleaning the world
Enhancements	E3	E1	E2

1.1.2 Team Video Link

<https://drive.google.com/file/d/1XGCHKvxCXyFYM-tzKQ7rC1j0PZlwhJqQ/view?usp=sharing>

1.1.3 Team Progress

1.1.3.1 Toms's Commits

- Created basic Menu structure and navigation
- Created Maze class and populated with setters/getters, data etc.
- Implemented Test Mode check
- Implemented Import Maze from terminal
- Implemented Random Maze Generation
- Implemented Testing Mode for Maze Generation

1.1.3.2 Liam's Commits

- Created test_invalidCharacterInput.input and expout files
- Created test_buildMaze.input and expout files
- Implemented functionality to find open blocks in maze to randomly teleport the player into
- Implemented findExitCoords() to get the exit coords of the maze for use in multiple functions
- Implemented maze solver using right-hand wall follower algorithm to show user the exit path
- Implemented testing mode for placing player in maze and solving maze using RHWF algorithm
- (E1) Implemented functionality to generate random mazes which take pre-existing terrain into account when generating their structure.
- (E1) Altered "Solve maze manually" and "Show Escape Route" functions to account for uneven terrain

1.1.3.3 Erfan's Commits

- Created test_generateMaze.input and expout file
- Created Build_Agent to contain type int values
- Created Block_Array::Vector to contain type int values
- Refactored Block_Array::Vector to use template for general data types
- Refactored Save_Terrain() to use getBlocks for heap connection + speed
- Implemented breadth first search for enhancement (E2) in my fork
- Implemented test for find shortest path enhancement in the fork
- Changed Restore_Terarin() to clear maze floor first to prevent falling sand or gravel blocks

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Agent	7
Block	7
Maze	8
Place_Maze	8
Solve_Maze	10
Terrain_Array	11

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Agent.h	13
Block.h	13
Maze.h	13
menuUtils.h	14
Place_Maze.h	16
Solve_Maze.h	17
Terrain_Array.h	18

Chapter 4

Class Documentation

4.1 Agent Struct Reference

Public Member Functions

- **Agent** (mcpp::Coordinate startLoc)

The documentation for this struct was generated from the following files:

- Agent.h
- Agent.cpp

4.2 Block Struct Reference

```
#include <Block.h>
```

Public Attributes

- int **x** = DEFAULT_X
- int **y** = DEFAULT_Y
- int **z** = DEFAULT_Z
- mcpp::BlockType **block_type** = DEFAULT_BLOCK

4.2.1 Detailed Description

Represents a block with coordinates and type

The documentation for this struct was generated from the following file:

- Block.h

4.3 Maze Class Reference

Public Member Functions

- **Maze** (unsigned int xLen, unsigned int zLen, bool mode)
- **Maze** (std::vector< std::vector< char > > mazeVec)
- void **Set_Len** (unsigned int xLen, unsigned int zLen)
- std::vector< unsigned int > **Get_Len** ()
- void **Set_Mode** (bool mode)
- bool **Get_Mode** ()
- void **Set_Maze** (std::vector< std::vector< char > > mazeVec)
- std::vector< std::vector< char > > **Get_Maze** ()
- void **Generate** ()
- void **Print** ()
- void **Print** (std::vector< std::vector< char > > mazeVec)
- unsigned int **getXLen** ()
- unsigned int **getZLen** ()

The documentation for this class was generated from the following files:

- Maze.h
- Maze.cpp

4.4 Place_Maze Class Reference

```
#include <Place_Maze.h>
```

Public Member Functions

- bool [Has_Terrain](#) ()
- void [Build_Maze](#) ()
- void [Flatten_Terrain](#) ()
- void [Load_Maze](#) (std::vector< std::vector< char > > maze_vec, int length, int width)
- void [Restore_Terrain](#) ()
- void [Save_Terrain](#) ()
- mcpp::Coordinate [Get_Player_Build_Pos](#) ()
- void [Set_Player_Build_Pos](#) (mcpp::Coordinate player_pos)
- void [Clear_Maze](#) ()
- void [Print_Maze_Size](#) ()
- void [Place_Carpet](#) ()

4.4.1 Detailed Description

Manages the placement and restoration of a maze

4.4.2 Member Function Documentation

4.4.2.1 Build_Maze()

```
void Place_Maze::Build_Maze ()
```

Builds the maze

4.4.2.2 Clear_Maze()

```
void Place_Maze::Clear_Maze ()
```

Clears the maze

4.4.2.3 Flatten_Terrain()

```
void Place_Maze::Flatten_Terrain ()
```

Flattens the terrain

4.4.2.4 Get_Player_Build_Pos()

```
mcpp::Coordinate Place_Maze::Get_Player_Build_Pos ()
```

Gets the player's build position

4.4.2.5 Has_Terrain()

```
bool Place_Maze::Has_Terrain ()
```

Checks if terrain is available

4.4.2.6 Load_Maze()

```
void Place_Maze::Load_Maze (
    std::vector< std::vector< char > > maze_vec,
    int length,
    int width)
```

Loads the maze from a vector

4.4.2.7 Place_Carpet()

```
void Place_Maze::Place_Carpet ()
```

Places carpet in the maze

4.4.2.8 Print_Maze_Size()

```
void Place_Maze::Print_Maze_Size ()
```

Prints the maze size

4.4.2.9 Restore_Terrain()

```
void Place_Maze::Restore_Terrain ()
```

Restores the original terrain

4.4.2.10 Save_Terrain()

```
void Place_Maze::Save_Terrain ()
```

Saves the current terrain

4.4.2.11 Set_Player_Build_Pos()

```
void Place_Maze::Set_Player_Build_Pos (
    mcpp::Coordinate player_pos)
```

Sets the player's build position

The documentation for this class was generated from the following files:

- Place_Maze.h
- Place_Maze.cpp

4.5 Solve_Maze Class Reference

Static Public Member Functions

- static bool **isOutsideMaze** (int &, const std::vector< mcpp::Coordinate > &)
- static mcpp::Coordinate **findExitCoords** (int, int, mcpp::Coordinate)
- static bool **placePlayerInMaze** (const std::vector< std::vector< char > > &, mcpp::Coordinate, int, int, bool)
- static bool **showEscapeRoute** (int, int, mcpp::Coordinate, const std::vector< std::vector< char > > &, bool)
- static void **getOpenBlocks** (const std::vector< std::vector< char > > &, mcpp::Coordinate, int, int)

The documentation for this class was generated from the following files:

- Solve_Maze.h
- Solve_Maze.cpp

4.6 Terrain_Array Class Reference

```
#include <Terrain_Array.h>
```

Public Member Functions

- **Terrain_Array** (const [Terrain_Array](#) &)=delete
- **Terrain_Array** & **operator=** (const [Terrain_Array](#) &)=delete
- void [add_block](#) (int x, int y, int z, mcpp::BlockType block)
- [Block](#) [get_block](#) (int index) const
- int [size](#) () const
- [Block](#) * **begin** () const
- [Block](#) * **end** () const
- void [print](#) () const
- void [clear](#) ()

4.6.1 Detailed Description

Manages a dynamic array of Blocks

4.6.2 Member Function Documentation

4.6.2.1 add_block()

```
void Terrain_Array::add_block (  
    int x,  
    int y,  
    int z,  
    mcpp::BlockType block)
```

Adds a block at the specified coordinates

4.6.2.2 clear()

```
void Terrain_Array::clear ()
```

Clears all blocks

4.6.2.3 get_block()

```
Block Terrain_Array::get_block (  
    int index) const
```

Retrieves a block by index

4.6.2.4 print()

```
void Terrain_Array::print () const
```

Prints all blocks

4.6.2.5 size()

```
int Terrain_Array::size () const
```

Returns the number of blocks

The documentation for this class was generated from the following files:

- Terrain_Array.h
- Terrain_Array.cpp

Chapter 5

File Documentation

5.1 Agent.h

```
00001 #ifndef AGENT_H
00002 #define AGENT_H
00003
00004 #include <mcpp/mcpp.h>
00005
00006 #define MOVE_XPLUS mcpp::Coordinate(1, 0, 0)
00007 #define MOVE_XMINUS mcpp::Coordinate(-1, 0, 0)
00008 #define MOVE_ZPLUS mcpp::Coordinate(0, 0, 1)
00009 #define MOVE_ZMINUS mcpp::Coordinate(0, 0, -1)
00010
00011 enum solveAlgorithm {
00012     RIGHT_HAND_FOLLOW,
00013     BREATH_FIRST_SEARCH,
00014 };
00015
00016 enum AgentOrientation { X_PLUS, Z_PLUS, X_MINUS, Z_MINUS };
00017
00018 struct Agent {
00019     Agent(mcpp::Coordinate startLoc);
00020     ~Agent();
00021 };
00022
00023 #endif // AGENT_H
```

5.2 Block.h

```
00001 #ifndef BLOCK_ARRAY_H
00002 #define BLOCK_ARRAY_H
00003
00004 #include <mcpp/block.h>
00005
00006 #define DEFAULT_X 0
00007 #define DEFAULT_Y 0
00008 #define DEFAULT_Z 0
00009 #define DEFAULT_BLOCK mcpp::Blocks::AIR
00010
00011 struct Block {
00012     int x = DEFAULT_X;
00013     int y = DEFAULT_Y;
00014     int z = DEFAULT_Z;
00015     mcpp::BlockType block_type = DEFAULT_BLOCK;
00016 };
00017
00018 #endif // BLOCK_ARRAY_H
```

5.3 Maze.h

```
00001 #ifndef ASSIGN_MAZE_H
00002 #define ASSIGN_MAZE_H
```

```

00003
00004 #include <mcpp/mcpp.h>
00005 #include <vector>
00006
00007 // Maze class stores only information related to construction or generation of
00008 // the maze, not the maze's location in minecraft
00009 class Maze {
00010 public:
00011     // Constructors
00012     Maze(unsigned int xLen, unsigned int zLen, bool mode);
00013     Maze(std::vector<std::vector<char>> mazeVec);
00014     Maze();
00015
00016     void Set_Len(unsigned int xLen, unsigned int zLen);
00017     std::vector<unsigned int> Get_Len();
00018
00019     void Set_Mode(bool mode);
00020     bool Get_Mode();
00021
00022     void Set_Maze(std::vector<std::vector<char>> mazeVec);
00023     std::vector<std::vector<char>> Get_Maze();
00024
00025     void Generate();
00026
00027     void Print();
00028     void Print(std::vector<std::vector<char>> mazeVec);
00029
00030     unsigned int getXLen();
00031     unsigned int getZLen();
00032
00033     // Destructors
00034     ~Maze();
00035
00036 private:
00037     // xLen = columns
00038     unsigned int xLen;
00039     // zLen = rows
00040     unsigned int zLen;
00041     bool mode;
00042     std::vector<std::vector<char>> mazeVec;
00043
00044     // This Function checks adjacent squares for a specific character
00045     void Check_Adj(std::vector<std::vector<char>>& mazeVec, unsigned int& xCurr,
00046                   unsigned int& zCurr, char targetChar, unsigned int dist,
00047                   bool& north, bool& east, bool& south, bool& west);
00048 };
00049
00050 #endif
00051 // ASSIGN_MAZE_H

```

5.4 menuUtils.h

```

00001 #include <iostream>
00002
00003 //TODO: Move functions to menuUtils.cpp
00004 void printStartText(void){
00005     std::cout << std::endl;
00006     std::cout << "Welcome to MineCraft MazeRunner!" << std::endl;
00007     std::cout << "-----" << std::endl;
00008 }
00009
00010 void printMainMenu(void){
00011     std::cout << std::endl;
00012     std::cout << "----- MAIN MENU -----" << std::endl;
00013     std::cout << "1) Generate Maze" << std::endl;
00014     std::cout << "2) Build Maze in MineCraft" << std::endl;
00015     std::cout << "3) Solve Maze" << std::endl;
00016     std::cout << "4) Show Team Information" << std::endl;
00017     std::cout << "5) Exit" << std::endl;
00018     std::cout << std::endl;
00019     std::cout << "Enter Menu item to continue: " << std::endl;
00020 }
00021
00022 void printGenerateMazeMenu(void){
00023     std::cout << std::endl;
00024     std::cout << "----- GENERATE MAZE -----" << std::endl;
00025     std::cout << "1) Read Maze from terminal" << std::endl;
00026     std::cout << "2) Generate Random Maze" << std::endl;
00027     std::cout << "3) Back" << std::endl;
00028     std::cout << std::endl;
00029     std::cout << "Enter Menu item to continue: " << std::endl;
00030 }
00031

```

```

00032 void printGenerateMazePrompt(void){
00033     std::cout << "In Minecraft, navigate to where you need the maze" << std::endl << "to be built in
        Minecraft and type - done:";
00034     std::cout << std::endl;
00035 }
00036
00037
00038 void printEnterLW(void){
00039     std::cout << "Enter the length and width of maze:";
00040     std::cout << std::endl;
00041 }
00042
00043 void printEnterStruct(void){
00044     std::cout << "Enter the maze structure:";
00045     std::cout << std::endl;
00046 }
00047
00048 void printMazeReadSuccess(void) {
00049     std::cout << "Maze read successfully";
00050     std::cout << std::endl;
00051 }
00052
00053 void printPrintingStart(void) {
00054     std::cout << "***Printing Maze***";
00055     std::cout << std::endl;
00056 }
00057
00058 void printPrintingEnd(void){
00059     std::cout << "***End Printing Maze***";
00060     std::cout << std::endl;
00061 }
00062
00063 void printReadyToSolve(void){
00064     std::cout << "Maze ready to Solve...";
00065     std::cout << std::endl;
00066 }
00067
00068 void printSolveMazeMenu(void){
00069     std::cout << std::endl;
00070     std::cout << "----- SOLVE MAZE -----" << std::endl;
00071     std::cout << "1) Solve Manually" << std::endl;
00072     std::cout << "2) Show Escape Route" << std::endl;
00073     std::cout << "3) Back" << std::endl;
00074     std::cout << std::endl;
00075     std::cout << "Enter Menu item to continue: " << std::endl;
00076 }
00077
00078
00079 void printTeamInfo(void){
00080     std::cout << std::endl;
00081     std::cout << "Team members:" << std::endl;
00082     std::cout << "\t [1] Erfan Samandarian (s4089117@student.rmit.edu.au)" << std::endl;
00083     std::cout << "\t [2] Liam Moore (s4095280@student.rmit.edu.au)" << std::endl;
00084     std::cout << "\t [3] Thomas Castanelli (s4072172@student.rmit.edu.au)" << std::endl;
00085     std::cout << std::endl;
00086 }
00087
00088 void printExitMessage(void){
00089     std::cout << std::endl;
00090     std::cout << "The End!" << std::endl;
00091     std::cout << std::endl;
00092 }
00093
00094 void printErrorMessageException(void){
00095     std::cout << "Error: Unknown Exception";
00096     std::cout << std::endl;
00097 }
00098
00099 void printErrorMessageInput(std::string errorStr) {
00100     std::cout << "Input Error: " << errorStr;
00101     std::cout << std::endl;
00102 }
00103
00104 void printErrorMessageRange(int in1, int in2){
00105     std::cout << "Input Error: Enter a number between ";
00106     std::cout << in1;
00107     std::cout << " and ";
00108     std::cout << in2;
00109     std::cout << " ...";
00110     std::cout << std::endl;
00111 }
00112
00113 // error if entered length and width (i.e. 3 and 5) are different to entered maze dimensions.
00114 void printErrorMessageArraySizeIncorrect(int length, int width) {
00115     std::cout << "Input Error: Entered length (" << length << ") and width (" << width << ") differs from
        manually entered maze's dimensions";
00116     std::cout << std::endl;

```

```

00117 }
00118
00119 void printErrorMessageNoMaze() {
00120     std::cout << "Input Error: No available maze to solve";
00121     std::cout << std::endl;
00122 }
00123
00124 void printErrorMessageInvalidMazeSize() {
00125     std::cout << "Input Error: At least one maze dimension has not been entered";
00126     std::cout << std::endl;
00127 }
00128
00129 void printErrorPlayerNotInMaze() {
00130     std::cout << "Input Error: Player is not in maze";
00131     std::cout << std::endl;
00132 }
00133
00134 // probably won't ever happen but eh
00135 void printErrorMazeNoSpace() {
00136     std::cout << "Input Error: Maze has no open space";
00137     std::cout << std::endl;
00138 }
00139
00140 void printBasePoint(mcpp::Coordinate playerPos) {
00141     std::cout << "BasePoint: " << playerPos << std::endl;
00142 }

```

5.5 Place_Maze.h

```

00001 #ifndef PLACE_MAZE_H
00002 #define PLACE_MAZE_H
00003
00004 #include "Solve_Maze.h"
00005 #include "Terrain_Array.h"
00006
00007 #include <mcpp/mcpp.h>
00008
00009 #include <algorithm>
00010 #include <chrono>
00011 #include <iostream>
00012 #include <iterator>
00013 #include <thread>
00014 #include <vector>
00015
00017 const std::vector<mcpp::BlockType> odd_blocks = {mcpp::Blocks::AIR,
00018     mcpp::Blocks::STILL_WATER,
00019     mcpp::Blocks::STILL_LAVA,
00020     mcpp::Blocks::FLOWING_WATER,
00021     mcpp::Blocks::FLOWING_LAVA,
00022     mcpp::Blocks::LILY_PAD,
00023     mcpp::Blocks::TALL_GRASS,
00024     mcpp::Blocks::ACACIA_LEAVES,
00025     mcpp::Blocks::ACACIA_SAPLING,
00026     mcpp::Blocks::ACTIVATOR_RAIL,
00027     mcpp::Blocks::BIRCH_LEAVES,
00028     mcpp::Blocks::BIRCH_SAPLING,
00029     mcpp::Blocks::CAKE_BLOCK,
00030     mcpp::Blocks::COBWEB,
00031     mcpp::Blocks::DARK_OAK_LEAVES,
00032     mcpp::Blocks::DARK_OAK_SAPLING,
00033     mcpp::Blocks::DEAD_BUSH,
00034     mcpp::Blocks::DEAD_SHRUB,
00035     mcpp::Blocks::DETECTOR_RAIL,
00036     mcpp::Blocks::FLOWER_POT,
00037     mcpp::Blocks::GRASS_PATH,
00038     mcpp::Blocks::JUNGLE_LEAVES,
00039     mcpp::Blocks::JUNGLE_SAPLING,
00040     mcpp::Blocks::LARGE_FERN,
00041     mcpp::Blocks::LILAC,
00042     mcpp::Blocks::OAK_LEAVES,
00043     mcpp::Blocks::OAK_SAPLING,
00044     mcpp::Blocks::POWERED_RAIL,
00045     mcpp::Blocks::RAIL,
00046     mcpp::Blocks::ROSE_BUSH,
00047     mcpp::Blocks::SPRUCE_LEAVES,
00048     mcpp::Blocks::SUNFLOWER};
00049
00051 class Place_Maze {
00052 public:
00053     Place_Maze() {};
00054
00056     bool Has_Terrain();
00058     void Build_Maze();

```

```

00060     void Flatten_Terrain();
00062     void Load_Maze(std::vector<std::vector<char>> maze_vec, int length,
00063                   int width);
00065     void Restore_Terrain();
00067     void Save_Terrain();
00068
00070     mcpp::Coordinate Get_Player_Build_Pos();
00072     void Set_Player_Build_Pos(mcpp::Coordinate player_pos);
00073
00075     void Clear_Maze();
00077     void Print_Maze_Size();
00078
00080     void Place_Carpet();
00081
00082 private:
00083     mcpp::MinecraftConnection mc;
00084     mcpp::Coordinate player_position;
00085
00086     int length = 0;
00087     int width = 0;
00088
00089     Terrain_Array terrain;
00090     Terrain_Array maze;
00091
00092     mcpp::BlockType player_position_block;
00093
00094     /* Blocks for placement of carpet */
00095     mcpp::BlockType carpet_block;
00096     mcpp::BlockType one_below_carpet_block;
00097     mcpp::BlockType two_below_carpet_block;
00098     mcpp::BlockType one_above_carpet_block;
00099     mcpp::BlockType two_above_carpet_block;
00100
00101     /* Coordinates for placement of carpet */
00102     mcpp::Coordinate carpet_coord;
00103     mcpp::Coordinate one_below_carpet_coord;
00104     mcpp::Coordinate two_below_carpet_coord;
00105     mcpp::Coordinate one_above_carpet_coord;
00106     mcpp::Coordinate two_above_carpet_coord;
00107 };
00108
00109 #endif

```

5.6 Solve_Maze.h

```

00001 #ifndef SOLVE_MAZE_H
00002 #define SOLVE_MAZE_H
00003
00004 #include <chrono>
00005 #include <iostream>
00006 #include <mcpp/mcpp.h>
00007 #include <thread>
00008
00009 class Solve_Maze {
00010 public:
00011     // returns true if solver reaches exit, and stops solver
00012     static bool isOutsideMaze(int&, const std::vector<mcpp::Coordinate>&);
00013
00014     // used to find the coordinates of the exit using the length and width of
00015     // the maze and checking for openings on the edges
00016     static mcpp::Coordinate findExitCoords(int, int, mcpp::Coordinate);
00017
00018     static bool placePlayerInMaze(const std::vector<std::vector<char>>&,
00019                                  mcpp::Coordinate, int, int, bool);
00020
00021     // option [2], guides the player out of the maze
00022     static bool showEscapeRoute(int, int, mcpp::Coordinate,
00023                                 const std::vector<std::vector<char>>&, bool);
00024
00025     // populated openCoordsField with open blocks within the maze
00026     static void getOpenBlocks(const std::vector<std::vector<char>>&,
00027                               mcpp::Coordinate, int, int);
00028
00029 private:
00030     // fields
00031     static mcpp::Coordinate exitCoords;
00032     static std::vector<mcpp::Coordinate> openCoordsField;
00033 };
00034
00035 #endif // SOLVE_MAZE_H

```

5.7 Terrain_Array.h

```
00001 #ifndef TERRAIN_ARRAY_H
00002 #define TERRAIN_ARRAY_H
00003
00004 #include "Block.h"
00005
00006 #include <iostream>
00007 #include <stdexcept>
00008
00010 class Terrain_Array {
00011 public:
00012     Terrain_Array();
00013     ~Terrain_Array();
00014
00015     Terrain_Array(const Terrain_Array&) = delete;
00016     Terrain_Array& operator=(const Terrain_Array&) = delete;
00017
00019     void add_block(int x, int y, int z, mcpp::BlockType block);
00021     Block get_block(int index) const;
00023     int size() const;
00024
00025     Block* begin() const { return blocks; }
00026     Block* end() const { return blocks + block_count; }
00027
00029     void print() const;
00030
00032     void clear();
00033
00034 private:
00036     void resize();
00037     Block* blocks;
00038     int block_count;
00039     int capacity;
00040 };
00041
00042 #endif // TERRAIN_ARRAY_H
```

Index

- add_block
 - Terrain_Array, [11](#)
- Agent, [7](#)
- Block, [7](#)
- Build_Maze
 - Place_Maze, [9](#)
- clear
 - Terrain_Array, [11](#)
- Clear_Maze
 - Place_Maze, [9](#)
- Flatten_Terrain
 - Place_Maze, [9](#)
- get_block
 - Terrain_Array, [11](#)
- Get_Player_Build_Pos
 - Place_Maze, [9](#)
- Has_Terrain
 - Place_Maze, [9](#)
- Load_Maze
 - Place_Maze, [9](#)
- Maze, [8](#)
- Place_Carpet
 - Place_Maze, [9](#)
- Place_Maze, [8](#)
 - Build_Maze, [9](#)
 - Clear_Maze, [9](#)
 - Flatten_Terrain, [9](#)
 - Get_Player_Build_Pos, [9](#)
 - Has_Terrain, [9](#)
 - Load_Maze, [9](#)
 - Place_Carpet, [9](#)
 - Print_Maze_Size, [9](#)
 - Restore_Terrain, [10](#)
 - Save_Terrain, [10](#)
 - Set_Player_Build_Pos, [10](#)
- print
 - Terrain_Array, [11](#)
- Print_Maze_Size
 - Place_Maze, [9](#)
- README, [1](#)
- Restore_Terrain
 - Place_Maze, [10](#)
- Save_Terrain
 - Place_Maze, [10](#)
- Set_Player_Build_Pos
 - Place_Maze, [10](#)
- size
 - Terrain_Array, [12](#)
- Solve_Maze, [10](#)
- Terrain_Array, [11](#)
 - add_block, [11](#)
 - clear, [11](#)
 - get_block, [11](#)
 - print, [11](#)
 - size, [12](#)